# Dating Game

# Dennis Shasha

# Omniheurist Course

# Computer Science

## *Description*

Matchmaker M is trying to figure out what kind of person P wants to date. So, M arranges many dates for P. The architects will report to M how much P likes those dates (score between -1 and 1, where 1 is good and -1 is very bad). P's criteria for liking a date or not depend on the weights P gives to various attributes -- e.g. literary knowledge, ability to solve puzzles, and others. The weights may be positive or negative ranging from -1 to 1 and specified as decimal values having at most two digits to the right of the decimal point, e.g. 0.13 but not 0.134. In each response, P may modify 5% of the criteria (P may choose which ones) by 20% each with respect to the original weights. For example, if a chosen criterion has a weight of 0.4 in P's original criteria, then P can modify it to any value between 0.4 - (0.2*0.4) and 0.4 + (0.2*0.4). P does these modifications without seeing any information provided by M nor any of the history of M's candidates. (So modifications are always with respect to P's original choice of weights.) Also, in both the original setting of weights and after all modifications, P must ensure that the sum of P's positive weights must be 1 and the sum of the negative weights must be -1.

A candidate C has values for each of n (not more than 100) attributes -- each value lies between 0 and 1 inclusive and must have four or fewer digits of precision (without this constraint, there is a way to discover the sign of the weight that P ascribes to each attribute). Given the weights P has assigned to each attribute w1, ..., wn and the values C has for each attribute v1, ..., vn, the score P gives to C is simply the dot product of the two vectors. No candidate should get a score below -1 or above 1, but the ideal candidate will consist solely of 1s and 0s.

You will play two roles: P and M. As P you set up your weights meeting the constraints above and send them to the architect who will verify that the constraints are met. In every round after the first 20, P may also change the weights though without knowledge of the current or previous candidates suggested by M. In the first round, the architect will then generate 20 candidates with values between 0 and 1 and apply those against P's original weights. The architect will give both the description of those candidate's values and their scores to M. As M you manufacture up to 20 additional candidates meeting the candidate constraints and receive scores from the architect iteratively. If you receive a score of 1 (using the pre-modification weights) you stop and the architect records how many candidates you required. Otherwise, you receive the maximum score you obtained after your 20 candidates. The value of M is the lexicographic value (score, number of candidates) A player X beats player Y if X obtains a greater score or obtains the same score but with fewer candidates than Y does for X's P.

## Architecture Team

Receive weights for P from a file or socket, verify attributes of an ideal candidate and an anti-ideal one (both provided by the player who puts up P). Then the architect generates 20 random candidates and provides the candidate profile and their scores to the matchmaker. All scores will have four digits of precision (exactly). Then receive iteratively a sequence of candidates from the M player (as well as modification instructions from P) and send to M the resulting scores. Supervise time constraint on the M and P players. Keep track of how many candidates M has used if successful.

Here is the loop:

```
until matchmaker has completed 20 candidates or received a score of 1
matchmaker submits a candidate
P submits modification instructions (without knowing anything about the candidate)
architect returns the score to M and displays the candidates
end until
```

Matchmaker records score and number of candidates proposed by matchmaker.

Here please find Anjali Menon's applet implementation